AMENDMENTS TO THE CLAIMS

The listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Canceled)

2. (Canceled)

3. (Canceled)

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (Canceled)

10. (Canceled)

11. (Canceled)

12. (Canceled)

13. (Canceled)

14. (Canceled)

15. (Canceled)

16. (Canceled)

17. (Canceled)

18. (Canceled)

19. (Canceled)

20. (Canceled)

21. (New) A computer implemented method of performing just-in-time (JIT) compilations, said method comprising initializing a JIT compiler, the initializing including:

initializing a mapping table, wherein the mapping table is used to track JIT compilations performed by the JIT compiler;

initializing a special filesystem that is used for memory mapping JIT compiled code, wherein the special filesystem responds to requests received from an operating system by:

responding to a load request to load a page by writing one or more invalid opcodes to the page and by returning a first successful completion code to the operating system; and

responding to a write request to write one or more pages from memory to a nonvolatile storage device by returning a second successful return completion code to the operating system without writing the one or more pages to the nonvolatile storage device;

registering an error handler with the operating system, wherein the error handler is called by the operating system when the invalid opcode is encountered; and

creating a special file using the special filesystem, wherein the special file is memory mapped to an address space, and wherein the address space is used to store JIT compiled code resulting from the JIT compiler compiling a plurality of methods.

22. (New) The method of claim 21 further comprising:

detecting, by the operating system, an instruction that causes a page fault resulting from a branch to a non-loaded page of memory;

requesting the special filesystem to handle the page fault;

receiving, at the special filesystem, the request to handle the page fault, wherein the special filesystem responds by writing the one or more invalid opcodes to one or more first addresses within a first address range within the address space, the first address range corresponding to the non-loaded page, and the special filesystem returns the first successful completion code to the operating system;

restarting, by the operating system, the instruction that caused the page fault, the restarting resulting in an invalid opcode error due to the one or more invalid opcodes written by the special filesystem to the first addresses; and

calling, by the operating system, the registered error handler in response to the invalid opcode error, wherein the error handler operates by:

loading a plurality of instructions in the first address range within the address space, the loading overwriting the invalid opcodes stored a the first addresses; and

restarting the instruction that caused the page fault, the restarting resulting in execution of the loaded instructions.

23. (New) The method of claim 22 wherein the loading of the plurality of instructions by the error handler further comprises:

receiving a page address of the non-loaded page;

identifying a method name from a plurality of method names stored in the mapping table, wherein the identified method name corresponds to the page address; and

writing the plurality of instructions to the first address range, wherein the plurality of instructions correspond to a method corresponding to the identified method name, and wherein the writing overwrites the one or more invalid opcodes previously written to the one or more first addresses.

24. (New) The method of claim 23 further comprising:

determining whether the page fault occurred at the beginning of the method, wherein the writing of the plurality of instructions is performed by JIT compilation when the page fault does not occur at the beginning of the method;

analyzing usage statistics corresponding to the method in response to the page fault occurring at the beginning of the method;

in response to the analysis revealing a high usage of the method:

performing the JIT compilation, wherein the JIT compilation results in the writing of the plurality of instructions; and

in response to the analysis revealing a lower usage of the method:

loading bytecode instructions into the first address range; and

removing the identified method name from the mapping table.

25. (New) The method of claim 21 further comprising:

identifying, at a memory manager, a heap page to discard from memory, wherein the identified heap page is being managed by the special filesystem;

instructing the special filesystem to write the identified heap page to a nonvolatile storage device;

in response to receiving the instruction at the special filesystem, returning the second successful completion code from the special file system to the memory manager, wherein the special filesystem does nothing with the heap page after receiving the instruction and before returning the second successful completion code;

receiving, at the memory manager, the second successful completion code; and

discarding, by the memory manager, the heap page in response to receiving the second successful completion code.

26. (New) The method of claim 21 further comprising:

determining that additional address space is needed to store additional JIT compiled code, wherein the additional JIT compiled code corresponds to one or more additional methods, and wherein the mapping table lists each method currently stored in the special file along with each method's address range;

analyzing usage statistics corresponding to the JIT code currently stored in the special file;

identifying one or more methods corresponding to seldom used JIT code; and

reclaiming address space from the special file by removing an entry corresponding to each of the identified seldom used methods from the mapping table.

27. (New) The method of claim 26 further comprising:

in response to determining that the special file is too small to store the additional JIT compiled code:

creating a secondary special file using the special file system;

memory mapping the secondary special file;

receiving, from the operating system, a secondary address space resulting from the memory mapping; and

initializing a secondary mapping table used to track the additional JIT compiled code stored in the secondary special file.

28. An information handling system comprising:

one or more processors;

a memory accessible by the processors;

a nonvolatile storage device accessible by the processors that includes one or more Java executable images, the Java executable images including a read-only section;

a just-in-time (JIT) compilation tool, the tool including software code that, when executed by the processors, performs steps that include:

initializing a JIT compiler, the initializing including:

initializing a mapping table, wherein the mapping table is used to track JIT compilations performed by the JIT compiler;

initializing a special filesystem that is used for memory mapping JIT compiled code, wherein the special filesystem responds to requests received from an operating system by:

responding to a load request to load a page by writing one or more invalid opcodes to the page and by returning a first successful completion code to the operating system; and

responding to a write request to write one or more pages from memory to a nonvolatile storage device by returning a second successful return completion code to the operating system without

writing the one or more pages to the nonvolatile storage device;

registering an error handler with the operating system, wherein the error handler is called by the operating system when the invalid opcode is encountered; and

creating a special file using the special filesystem, wherein the special file is memory mapped to an address space, and wherein the address space is used to store JIT compiled code resulting from the JIT compiler compiling a plurality of methods.

29. (New) The information handling system of claim 28 wherein the software code, when executed by the processors, performs further steps comprising:

detecting, by the operating system, an instruction that causes a page fault resulting from a branch to a non-loaded page of memory;

requesting the special filesystem to handle the page fault;

receiving, at the special filesystem, the request to handle the page fault, wherein the special filesystem responds by writing the one or more invalid opcodes to one or more first addresses within a first address range within the address space, the first address range corresponding to the non-loaded page, and the special filesystem returns the first successful completion code to the operating system;

restarting, by the operating system, the instruction that caused the page fault, the restarting resulting in an invalid opcode error due to the one or more invalid opcodes written by the special filesystem to the first addresses; and

calling, by the operating system, the registered error handler in response to the invalid opcode error, wherein the error handler operates by:

loading a plurality of instructions in the first address range within the address space, the loading overwriting the invalid opcodes stored a the first addresses; and

restarting the instruction that caused the page fault, the restarting resulting in execution of the loaded instructions.

30. (New) The information handling system of claim 29 wherein the loading of the plurality of instructions by the error handler further comprises:

receiving a page address of the non-loaded page;

identifying a method name from a plurality of method names stored in the mapping table, wherein the identified method name corresponds to the page address; and

writing the plurality of instructions to the first address range, wherein the plurality of instructions correspond to a method corresponding to the identified method name, and wherein the writing overwrites the one or more invalid

opcodes previously written to the one or more first addresses.

31. (New) The information handling system of claim 30 wherein the software code, when executed by the processors, performs further steps comprising:

determining whether the page fault occurred at the beginning of the method, wherein the writing of the plurality of instructions is performed by JIT compilation when the page fault does not occur at the beginning of the method;

analyzing usage statistics corresponding to the method in response to the page fault occurring at the beginning of the method;

in response to the analysis revealing a high usage of the method:

performing the JIT compilation, wherein the JIT compilation results in the writing of the plurality of instructions; and

in response to the analysis revealing a lower usage of the method:

loading bytecode instructions into the first address range; and

removing the identified method name from the mapping table.

32. (New) The information handling system of claim 28 wherein the software code, when executed by the processors, performs further steps comprising:

identifying, at a memory manager, a heap page to discard from memory, wherein the identified heap page is being managed by the special filesystem;

instructing the special filesystem to write the identified heap page to a nonvolatile storage device;

in response to receiving the instruction at the special filesystem, returning the second successful completion code from the special file system to the memory manager, wherein the special filesystem does nothing with the heap page after receiving the instruction and before returning the second successful completion code;

receiving, at the memory manager, the second successful completion code; and

discarding, by the memory manager, the heap page in response to receiving the second successful completion code.

33. (New) The information handling system of claim 28 wherein the software code, when executed by the processors, performs further steps comprising:

determining that additional address space is needed to store additional JIT compiled code, wherein the additional JIT compiled code corresponds to one or more additional methods, and wherein the mapping table lists each method

currently stored in the special file along with each method's address range;

analyzing usage statistics corresponding to the JIT code currently stored in the special file;

identifying one or more methods corresponding to seldom used JIT code;

reclaiming address space from the special file by removing an entry corresponding to each of the identified seldom used methods from the mapping table; and

in response to determining that the special file is too small to store the additional JIT compiled code:

creating a secondary special file using the special file system;

memory mapping the secondary special file;

receiving, from the operating system, a secondary address space resulting from the memory mapping; and

initializing a secondary mapping table used to track the additional JIT compiled code stored in the secondary special file.

34. (New) A computer program product that includes software code encoded in a computer readable medium that, when executed by an information handling system, causes the information handling system to perform steps comprising:

initializing a mapping table, wherein the mapping table is used to track JIT compilations performed by the JIT compiler;

initializing a special filesystem that is used for memory mapping JIT compiled code, wherein the special filesystem responds to requests received from an operating system by:

responding to a load request to load a page by writing one or more invalid opcodes to the page and by returning a first successful completion code to the operating system; and

responding to a write request to write one or more pages from memory to a nonvolatile storage device by returning a second successful return completion code to the operating system without writing the one or more pages to the nonvolatile storage device;

registering an error handler with the operating system, wherein the error handler is called by the operating system when the invalid opcode is encountered; and

creating a special file using the special filesystem, wherein the special file is memory mapped to an address space, and wherein the address space is used to store JIT compiled code resulting from the JIT compiler compiling a plurality of methods.

35. (New) The computer program product of claim 34 wherein the software code causes the information handling system to perform further steps comprising:

detecting, by the operating system, an instruction that causes a page fault resulting from a branch to a non-loaded page of memory;

requesting the special filesystem to handle the page fault;

receiving, at the special filesystem, the request to handle the page fault, wherein the special filesystem responds by writing the one or more invalid opcodes to one or more first addresses within a first address range within the address space, the first address range corresponding to the non-loaded page, and the special filesystem returns the first successful completion code to the operating system;

restarting, by the operating system, the instruction that caused the page fault, the restarting resulting in an invalid opcode error due to the one or more invalid opcodes written by the special filesystem to the first addresses; and

calling, by the operating system, the registered error handler in response to the invalid opcode error, wherein the error handler operates by:

loading a plurality of instructions in the first address range within the address space, the loading overwriting the invalid opcodes stored a the first addresses; and

restarting the instruction that caused the page fault, the restarting resulting in execution of the loaded instructions.

36. (New) The computer program product of claim 35 wherein the software code that performs the loading of the plurality of instructions by the error handler performs further steps comprising:

receiving a page address of the non-loaded page;

identifying a method name from a plurality of method names stored in the mapping table, wherein the identified method name corresponds to the page address; and

writing the plurality of instructions to the first address range, wherein the plurality of instructions correspond to a method corresponding to the identified method name, and wherein the writing overwrites the one or more invalid opcodes previously written to the one or more first addresses.

37. (New) The computer program product of claim 36 wherein the software code causes the information handling system to perform further steps comprising:

determining whether the page fault occurred at the beginning of the method, wherein the writing of the plurality of instructions is performed by JIT compilation when the page fault does not occur at the beginning of the method;

analyzing usage statistics corresponding to the method in response to the page fault occurring at the beginning of the method;

in response to the analysis revealing a high usage of the method:

performing the JIT compilation, wherein the JIT compilation results in the writing of the plurality of instructions; and

in response to the analysis revealing a lower usage of the method:

loading bytecode instructions into the first address range; and

removing the identified method name from the mapping table.

38. (New) The computer program product of claim 34 wherein the software code causes the information handling system to perform further steps comprising:

identifying, at a memory manager, a heap page to discard from memory, wherein the identified heap page is being managed by the special filesystem;

instructing the special filesystem to write the identified heap page to a nonvolatile storage device;

in response to receiving the instruction at the special filesystem, returning the second successful completion code from the special file system to the memory manager, wherein the special filesystem does nothing with the heap page after receiving the instruction and before returning the second successful completion code;

receiving, at the memory manager, the second successful completion code; and

discarding, by the memory manager, the heap page in response to receiving the second successful completion code.

39. (New) The computer program product of claim 34 wherein the software code causes the information handling system to perform further steps comprising:

determining that additional address space is needed to store additional JIT compiled code, wherein the additional JIT compiled code corresponds to one or more additional methods, and wherein the mapping table lists each method currently stored in the special file along with each method's address range;

analyzing usage statistics corresponding to the JIT code currently stored in the special file;

identifying one or more methods corresponding to seldom used JIT code; and

reclaiming address space from the special file by removing an entry corresponding to each of the identified seldom used methods from the mapping table.

40. (New) The computer program product of claim 39 wherein the software code causes the information handling system to perform further steps comprising:

in response to determining that the special file is too small to store the additional JIT compiled code:

creating a secondary special file using the special file system;

memory mapping the secondary special file;

receiving, from the operating system, a secondary address space resulting from the memory mapping; and

initializing a secondary mapping table used to track the additional JIT compiled code stored in the secondary special file.